NAMRL Special Report 94-1

# THE SHARPER IMAGE: IMPLEMENTING A FAST FOURIER TRANSFORM (FFT) TO ENHANCE A VIDEO-CAPTURED IMAGE

J. E. Parker, W. K. Krebs, J. S. Marsh, D. L. Still, and L. A. Temme



Naval Aerospace Medical Research Laboratory
Naval Air Station
Pensacola, Florida 32508-5700

NAVAL AEROSPACE MEDICAL RESEARCH LABORATORY
51 HOVEY ROAD, PENSACOLA, FL 32508-1046

NAMRL Special Report 94-1

# THE SHARPER IMAGE: IMPLEMENTING A FAST FOURIER TRANSFORM (FFT) TO ENHANCE A VIDEO-CAPTURED IMAGE

J. E. Parker, W. K. Krebs, J. S. Marsh,[1]
D. L. Still, and L. A. Temme

[1]University of West Florida
Department of Physics
Pensacola, FL 32514

Reviewed and approved 21 Jan 94

A. J. Mateczun

A. J. MATECZUN, CAPT, MC USN
Commanding Officer


This research was sponsored by the Naval Medical Research and Development Command under work unit 62233N MM33130.009-7305.

The views expressed in this article are those of the authors and do not reflect the official policy or position of the Department of the Navy, Department of Defense, nor the U.S. Government.

Trade names of materials and/or products of commercial or nongovernment organizations are cited as needed for precision. These citations do not constitute official endorsement or approval of the use of such commercial materials and/or products.

## ABSTRACT

The lack of visual information and cues in night vision images has produced a history of in-flight mishaps where night vision devices (NVDS) were used as the primary source of sight. In an effort to produce improved night vision images and thereby reduce in-flight mishaps, a video enhancement technique, the sharper image, was developed to enhance, filter, and study images generated by NVDS. The sharper image technique enhances night vision images by filtering or amplifying the spatial frequencies that are distorted by NVDS. In this report, we describe the necessary methodology and procedures to enhance, filter, or process any visual image. We also discuss other computational image software and review the advantages and disadvantages of each.

## Acknowledgments

The authors would like to acknowledge the following people for their assistance. A special thanks to Ms. Shirley Dasho for her assistance in the correct organizational format of the technical report and to Mr. Jim Paul for the excellent photographs.

# INTRODUCTION

The vision research team at the Naval Aerospace Medical Research Laboratory (NAMRL) has initiated the sharper image project to produce improved night vision images and thereby reduce in-flight mishaps. Since the Vietnam War, the ability of night vision devices (NVDs) to amplify ambient light has been used by naval aviators to aid them in night attack raids. However, simply increasing the amount of light that impinges on the retina does not necessarily improve one's visual perception of the surroundings. Photochemical changes within the retinal receptors, transduction of neural activity along the visual pathway, and components within the brain that interpret this neural activity are also important to the development of visual perception. The lack of visual information and cues in night vision images has produced a history of in-flight mishaps where NVDs are used as the primary source of sight.

Still and Temme (1993) found that one cause of the lack of visual information derives from the suppression of spatial frequency contrasts across all frequencies with the effect being most pronounced on the lower spatial frequency bands. In their study, viewers were presented with a series of single spatial frequency test gratings. At each frequency tested, the contrast between the light and dark bands was reduced until the subject could no longer distinguish the test grating from a uniformly grey background. After the subject's threshold was determined using his naked eye, the subject donned a pair of night vision goggles and repeated the test. A photometer ensured that the light levels reaching the subject's eye were the same for both tests. The authors concluded that contrast sensitivity thresholds are significantly lower for the subjects wearing goggles. The difference between the aided and unaided thresholds was greatest at lower spatial frequencies. These results indicate that the light amplification process of the goggles is not effectively transmitting the spatial frequency signals into the images.

In another study, Peli, Goldstein, Young, Trempe, and Buzney (1991) found that most visually impaired persons fail to discern the higher spatial frequencies present in an image. Based on the Fourier analysis of vision, Peli et al. improved the vision of patients by using the Modulation Transfer Function (MTF) to identify spatial frequency components that are poorly transferred then proportionately amplifying these components of the input signal. By presenting the same images with the higher spatial frequencies enhanced above normal levels, their subjects were significantly more successful in identifying objects and faces.

We sought to develop a similar technique with NVDs and hopefully provide night attack pilots with the necessary visual cues to make their job safer. Our process uses a Macintosh computer equipped with a NuVista® + analog to digital (A/D) converter to analyze night vision images captured on CCTV or video tape. The NuVista® + interface accepts video input signals from CCTV, RGB, composite video, and S-video cameras and digitizes the signal into a 512 x 400 pixel array of greyscale intensity values. The arrays are loaded into the program *Mathematica*®, which performs the Fourier transforms and filtering tasks. Finally, the program returns the image to the spatial domain using the inverse Fourier transform, and the images are visually inspected to determine the effectiveness of that filtering process in producing a better quality image.

## FOURIER ANALYSIS

Since the NVD's responses vary over the input spatial frequency domain, one can enhance the NVD images by amplifying or reducing certain frequencies to compensate for the distortion caused by the NVD. To manipulate the spatial frequencies of an image, one must decompose the original image into its constituent spatial frequencies, amplify the frequencies of interest, and resynthesize the new image. The process is facilitated by the use of Fourier decomposition on the image.

The method of Fourier analysis provides a linear mathematical system to quantitatively analyze and compare complex wave forms. In 1807, Baron Jean-Baptiste-Joseph Fourier proved that any periodic wave can be represented by a superposition of several sine and cosine waves (DeValois and DeValois, 1988). The

Fourier transform (equation 1) of the spatial function f(x) provides information about the power spectrum, amplitude, and phase of each of the constituent sinusoidal waves that create the original complex wave form.

$$F(\alpha) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\alpha x} dx \tag{1}$$

Here, x is the location in space, $\alpha$ is the spatial frequency, and i is the imaginary root $\sqrt{-1}$ . Though the Fourier transform is a powerful analytical tool, it was not until Cooley and Tukey developed the Fast Fourier Transform (FFT) algorithm in 1965 that numerical calculations become feasible (Weaver, 1983). Modern FFT routines are still computation intensive, but most personal computers can perform the analysis in a reasonable time.

To calculate the FFT of an image, each picture is treated as a two-dimensional array of discrete greyscale intensities. This matrix is transformed using the two-dimensional analog of the one-dimensional Discrete Fourier Transform (Equation 2)

$$F'(k_1,k_2) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} V'(n_1,n_2) \exp\left[-2\pi i\left(\frac{k_1 n_1}{N_1} + \frac{k_2 n_2}{N_2}\right)\right] \tag{2}$$

Here, the integrals over the continuous spatial variable x have been replaced by finite sums of the discrete variables $n_1$ and $n_2$. The discrete nature of the Fourier Transform derives from the digitized nature of the computer image. Instead of continuous x and y coordinates of intensity, the picture has been broken down into pixels of finite size.

Once the images are transformed, each row and column of the array represents the amplitudes of the harmonically related spatial frequencies present in the image. Filtering out and amplifying specific spatial frequencies becomes simply a matter of locating an element in the array and multiplying it by the amplification factor. For example, to perform a low-frequency filter, which eliminates all components below a set threshold frequency, find the region in the array that corresponds to frequencies less than that threshold frequency and set those elements to zero. After the spatial components have been modified, the image can be recompiled for presentation using the two-dimensional Discrete Inverse Fourier Transform (Equation 3).

$$V'(n_1,n_2) = \frac{1}{N_1 N_2} \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} F'(k_1,k_2) \exp\left[2\pi i\left(\frac{k_1 n_1}{N_1} + \frac{k_2 n_2}{N_2}\right)\right] \tag{3}$$

# METHODS

## DATA COLLECTION

Images for the video-capture project were taken in the field with VHS, Super VHS video or 35-mm cameras that were modified with NVD attachments. The NVD was mounted in front of the camera lens, and a coupling connected the NVD eyepiece to the camera lens to shield outside lighting. Image quality varied with the camera. Images from 35-mm film exhibited the sharpest contrasts, and normal VHS images had the lowest resolution.

## DIGITIZATION

We converted the images into a computer-compatible format using a NuVista® + card and the application program *Capture* + (Truevision, 1991). To digitize VHS or Super VHS video images, the playback VCR's video output was connected to an Apple Macintosh Quadra 950 equipped with a NuVista® + card. For VHS images, only the signal from the composite video out was necessary. Super VHS images required four input signals to the NuVista® + card. The pin connections for the VHS and Super VHS inputs are shown below:

| Pin Number | Input Connector | Output Connector |
|---|---|---|
| 1 | Ground | Ground |
| 2 | Ground | Ground |
| 3 | Red input | Red output |
| 4 | Green input | Green output |
| 5 | Blue input | Blue output |
| 6 | Composite video input | Composite video output |
| 7 | Composite sync input | Composite sync output |
| 8 | Luma input (S-video) | Luma/H sync output |
| 9 | Chroma input (S-video) | Chroma/Vsync output |

To obtain better resolution of field images, a 35-mm photograph was transferred into the computer via a closed-circuit television camera. The pin connections for the closed-circuit television camera's video output signal were the same as the VHS signal.

Actual video capturing was performed by the application program *Capture* +. Images from the closed-circuit camera were the easiest to capture. To capture these images, we toggled the command "Capture Image," and once the image was positioned satisfactorily, clicked the mouse button again. The captured image was then converted into a PICT file format and saved.

Images from VHS or Super VHS recordings were captured using the same process as outlined above, but the tape must be running to capture these VHS or Super VHS recordings. However, because the images cannot be paused, several attempts were needed to get the image at the right time. Often the captured image had two horizontal lines in the picture--a consequence of capturing between frames of the tape. The only solution was to recapture the image until the lines were at the top and bottom, signifying that a complete frame had been captured. Once captured, the image was also saved in a PICT format. An example of a captured image from a closed-circuit camera is shown in Fig. 1.

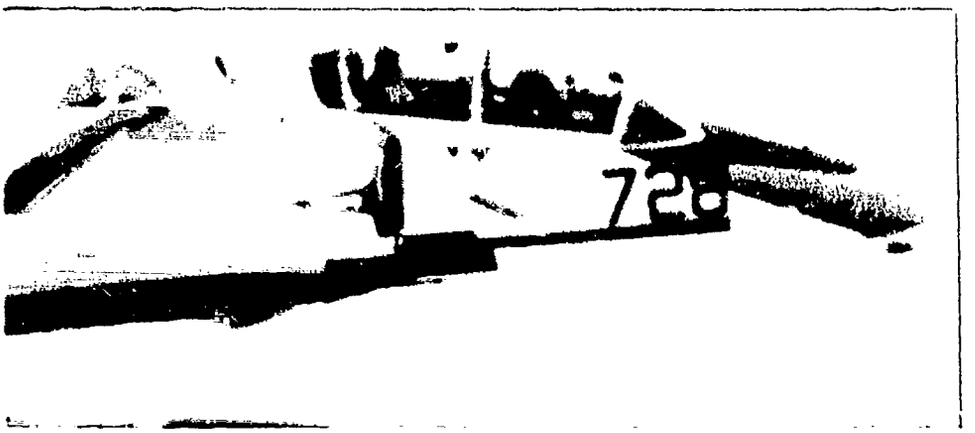

Figure 1. Image captured from a CCTV signal. This digitized image of a TA-4 will be used in all succeeding image enhancement techniques.

**FREQUENCY DOMAIN**

To execute the Fourier analysis, the digitized images must be in a readable format for the mathematics application program *Mathematica*®. Because *Mathematica*® can compute Fourier transforms of nonsquare matrices, restructuring the image as a $2^N$ square matrix is not necessary, although it will make the calculations faster. To prepare the image for manipulation with Mathematica®, we followed the steps outlined below:

Step 1: Select image by clicking the mouse on the right margin
Step 2: From the graph menu choose "Convert to input Form"
Step 3: Click OK from pop-up to execute
Step 4: Delete these highlighted sections from result

```
Show[Graphics[{
Raster[{
{0.5856, 0.5770, 0.5769,.....,
 .....,0.5244, 0.5287, 0.5323, 0.5307, 0.5381}},
{{0.000000, 0.000000}, {1.000000, 0.706587}}]
}], AspectRatio->0.70659,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.70659}}]
```

Step 5: Assign name to matrix and add a semicolon at the end of statement

```
image = {{0.5856, 0.5770, 0.5769,...}, {...,.....},{
 .....},{...,0.5244, 0.5287, 0.5323, 0.5307, 0.5381}};
```

The output matrix contains the amplitude of the sinusoidal functions that compose the Fourier series. An example of a resultant matrix is shown in Fig. 2. The headings of each column indicate the order of the harmonic frequency of that term.

| | 0 | +1 | +2 | ... | +M/2 | +1-M/2 | +2-M/2 | ... | -1 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | |
| +1 | | | | | | | | | |
| +2 | | | | | | | | | |
| ⋮ | | | | | | | | | |
| +N/2 | | | | | | | | | |
| +1-N/2 | | | | | | | | | |
| +2-N/2 | | | | | | | | | |
| ⋮ | | | | | | | | | |
| -1 | | | | | | | | | |

Figure 2. The format *Mathematica*® uses to store Fourier matrices. The column and row headings indicate the order of the harmonic term, M represents the number of pixels that the original image contained in the horizontal direction, and N represents the number of pixels in the vertical.

By default, *Mathematica*® organizes the power spectrum with the zero-frequency component in the upper left-hand quadrant. This format does not follow the standard power spectrum in previous computational vision literature, where the zero frequency component is almost always positioned in the center. An example of such a power spectrum is shown in Fig. 3.

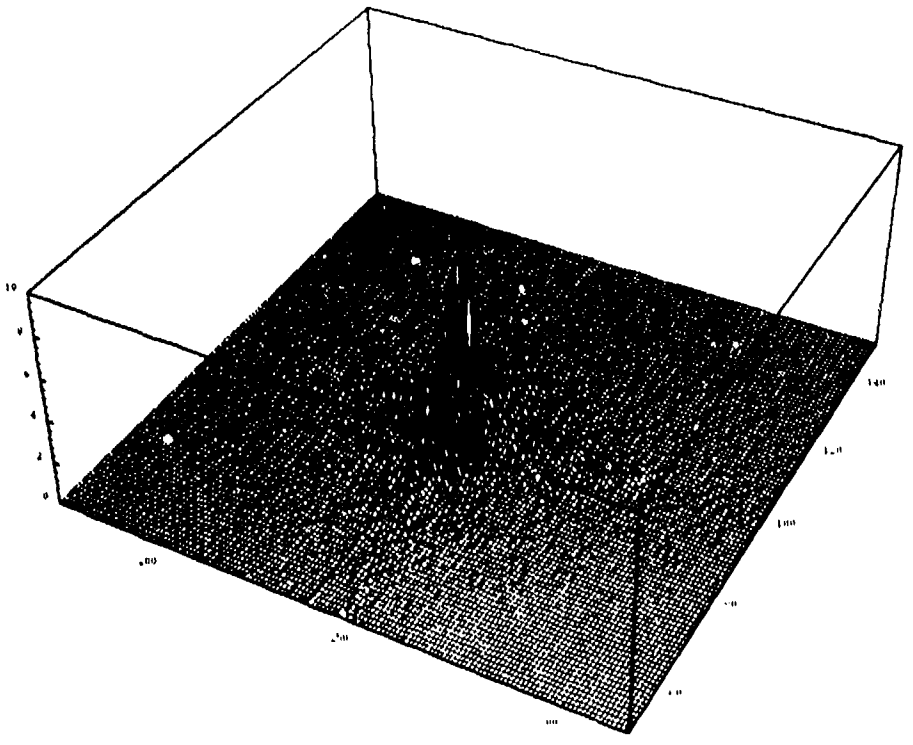

Figure 3. Power spectrum of A-4 plane with zero frequency at center.

To arrange the power spectrum as shown in Fig. 3, the programming code is described in Appendix A.

## FILTERING

For image processing, we used computer routines to selectively remove or enhance certain spatial frequency components. Several different filters were developed for the project: low- and high-frequency, band-pass, oblique, and low- and high- frequency amplifiers.

The low-frequency filter was designed such that the spatial frequency components above a set threshold were removed from the image, and any lower frequencies passed through the filter unaltered. Such filters are often used to remove random noise from an image because the noise is often characterized by very high spatial frequencies. Figure 4 shows the result of a low-pass filter that removed all frequency components higher than the 30th harmonic. Refer to Appendix B for the low-pass filter programming code.

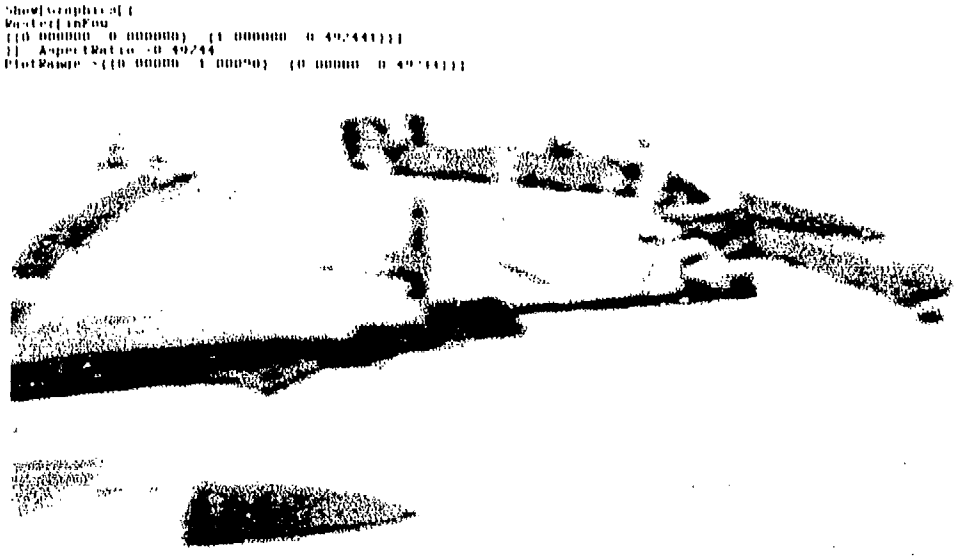

Figure 4. Result of applying a low-pass filter to the image of the TA-4. All frequency components higher than the 30th harmonic have been removed.

High-frequency filters operate similarly to low pass filters: all frequency components below a set threshold are removed, and the higher frequencies are passed. An example of high-pass filtering is demonstrated in Fig. 5. Here, all frequency components below the 30th harmonic have been removed. Refer to Appendix C for high-pass filter programming code.


Figure 5. Result of applying a high-pass filter to the image of the TA-4. All frequency components less than the 30th harmonic have been removed.

Combining the attributes of both high- and low-pass filters describes the design of a band-pass filter. The band pass filter allowed only the spatial frequency components within a pair of limits to pass through the filter; all others are were set to zero. This filter is demonstrated in Fig. 6, where the low threshold is set to be the 4th harmonic, and the high threshold is the 30th. Refer to Appendix D for the band-pass filter programming code. The high and low spatial frequency amplifiers were designed to amplify either the high or low spatial frequencies in the image without affecting the other frequencies. Figure 7 presents the result of a low spatial frequency amplifier on the image of the plane. For this image, the threshold frequency was set at the 30th harmonic, and each component in the lower range was amplified 50%. Figure 8 presents the result of a high spatial frequency amplifier. Here, the threshold frequency is set at the 30th harmonic, and each component in the higher range has been amplified 20%.

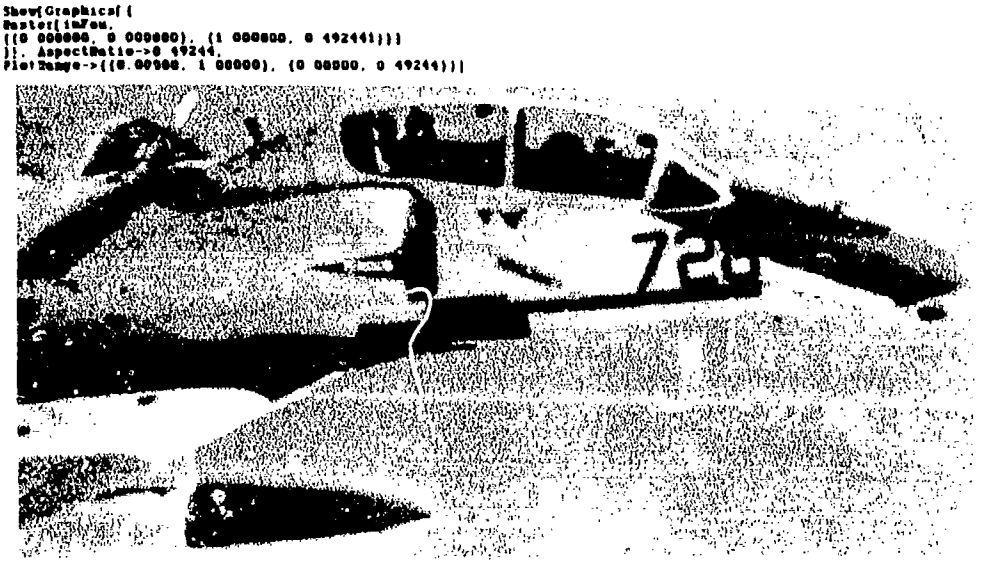


Figure 6. Result of applying a band-pass filter to the image of the TA-4. This band pass filter has been set to accept only frequency components between the 4th and 30th harmonics.

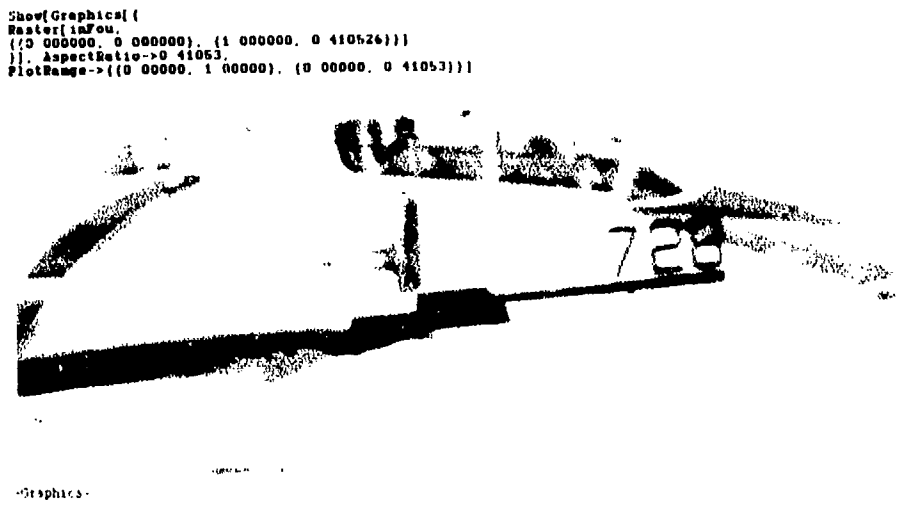


Figure 7. Result of applying a low spatial frequency amplifier to the image of the TA-4. The threshold frequency is set at the 30th harmonic, and each component below threshold has been amplified 50%.

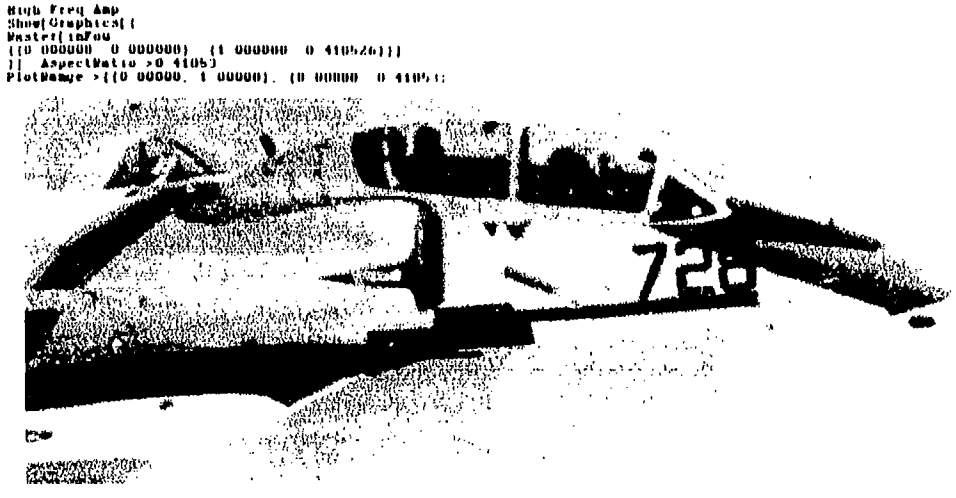


Figure 8. Result of applying a high spatial frequency amplifier to the image of the TA-4. The threshold frequency is set at the 30th harmonic, and each component above threshold has been amplified 20%. Refer to Appendix E for amplifying a low spatial frequency and Appendix F for amplifying a high spatial frequency.

The oblique filters were designed to highlight diagonal lines present in an image. By eliminating all elements of the Fourier matrix except those on the prime diagonal, the resynthesized image contained only the diagonal lines of the original. Figure 9 demonstrates the result of an 135° oblique filter. Refer to Appendix G for the oblique filter programming code.

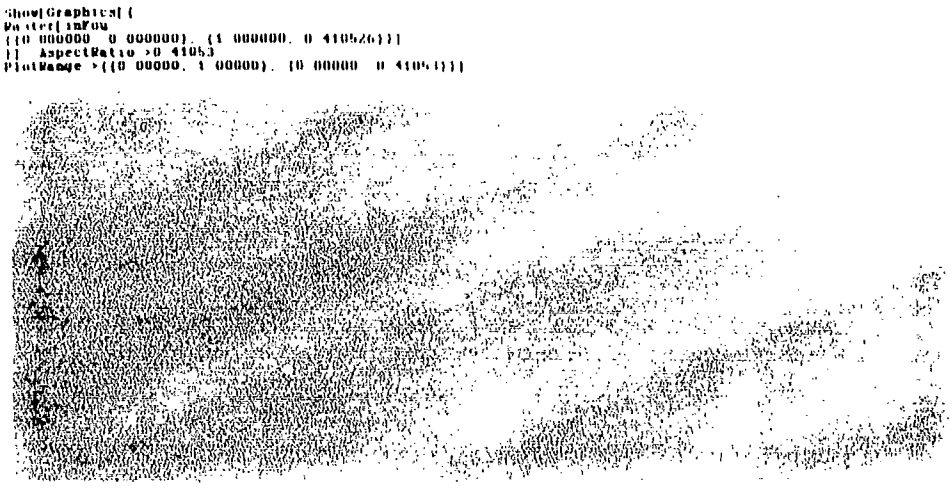

Figure 9. Result of applying a 135° oblique filter to the image of the TA-4.

On completion of the task, the computer generated a matrix **inFou** that was the resultant of the filtering process and an image that was the result of the Discrete Inverse Fourier transform performed on the filtered matrix. The matrix variable **inFou** was saved to disk for use in further filtering processes.

## DISCUSSION AND CONCLUSIONS

The *Mathematica*® program was very versatile tool for image processing. Its ability to handle any size array and programming language support allowed this application to be custom designed to fit our need. Two other Fourier packages on the market, Image 2.86 (Reeves, 1990) and *Transform* (Spyglass, 1991), were compared for capabilities similar to those of *Mathematica*®. *Image 2.86* is a menu-driven application in which filtering routines can be used to perform most of the filtering processes described. Its drawbacks include 1) images are forced to be $2^N$ pixels square; 2) amplification of spatial frequencies is not supported; and 3) because it is not possible to directly manipulate the matrix, filters that depend on the position of the element in the matrix (e.g., the oblique filter) are not supported either. However, *Image 2.86* incorporates an additional feature, a user-definable transition zone, into its filtering routines. *Mathematica*® currently acts as a step filter; a spatial frequency is either within the filter's limits and gets passed 100%, or outside the limits and is removed 100%. *Image 2.86* has defined cosine, gaussian, linear and quadratic transition zones that allow a variable percentage of the spatial frequencies that fall within the transition zone to pass. Figure 10 demonstrates the variations between these transition zones.

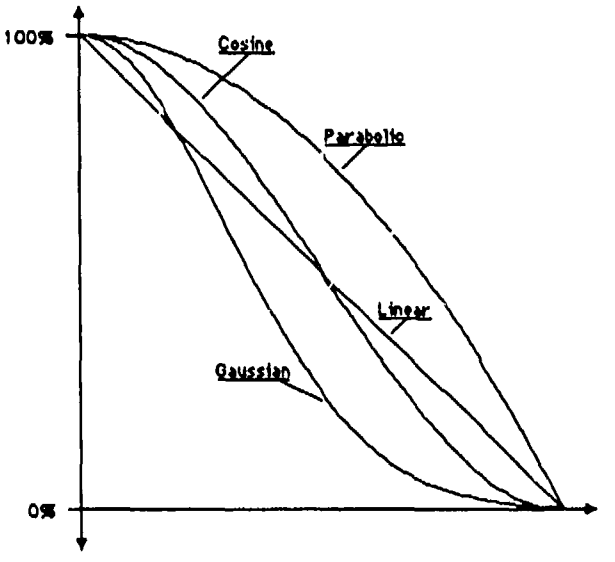


Figure 10. Variations between transition zones defined in *Image 2.86* (Reeves, 1990).

Although this feature is not included in the *Mathematica*® programs, a subroutine can be added that will enable the application to perform this task.

The application *Transform* operates very similarly to *Mathematica*®. Both applications rely on written computer code to filter the image. Each has its trade-offs. *Transform* accepts C and FORTRAN programs, which allows it to incorporate preexisting libraries of computer code into its filters, but it is limited to images that are $2^N$ square pixels. *Mathematica*® accepts any image, but the filter routines are developed from the *Mathematica*® language, which is nonstandard.

One of the disappointments we encountered is the inability of the NuVista®+ card to capture still images from a VCR. When the tape is paused, the VCR does not transmit the image signal in standard National Television Standards Committee (NTSC) format, but rather shifts the frequency resolution up slightly. Standard television sets have a degree of "slop" in their interpretation of the incoming frequency and are able to cope with the change. Because the NuVista®+ card is very precise in its digitization sampling, the change in transmission format causes the signals to be misinterpreted. Thus, the captured image is of very poor quality. A solution is to feed the output of the VCR through a timebase corrector, which will return the signal to an NTSC format that NuVista® can recognize. The only problem with this solution is that timebase correctors are expensive. Another solution is to display the image via a normal television set and capture the image using a closed-circuit camera that has a direct feed to the NuVista® card. The problem here is that the image quality may be limited by the camera's resolution.

## SUMMARY

This technical report is intended to be an introductory tool for *Mathematica*® and NuVista®+. The programs outlined in the report are designed to give a naive *Mathematica*® user an opportunity to implement several different computational vision techniques for enhancing a visual image. The programs are developed for a specific NVD project at NAMRL and may not be appropriate for other projects. However, the report incorporates a wide variety of computational vision techniques that would be useful for image enhancement.

## REFERENCES


De Valois, R. L., & De Valois, K. K. (1988). Spatial vision. New York: Oxford University Press.

Peli, E., Goldstein, R. B., Young, G. M., Trempe, C. L., & Buzney, S. M. (1991). Image enhancement for the visually impaired. *Investigative Ophthalmology and Visual Science, 32*, 2337-2350.

Reeves, A. A. (1990). *Optimized fast Hartley transform for the MC68000 with applications in Image processing.* Unpublished master's thesis, Dartmouth College, Hanover, NH.

Spyglass®, Inc. (1991). *Transform version 2.0* [Computer program]. Champaign, IL: Spyglass®, Inc.

Still, D. L., and Temme, L. A., (1993). *The Modulation Transfer Function (MTF) of Night Vision Goggles (NVG).* Proceedings of the Aerospace Medical Association 64th Annual Scientific Meeting, Toronto, Canada (p. 427).

Truevision®, Inc. (1991). *NuVista®+ Capture* [Computer program]. Indianapolis, IN: Truevision®.

Weaver, H. J. (1983). *Applications of discrete and continuous Fourier analysis.* New York: John Wiley & Sons.

Wolfram, S. (1991). *Mathematica®: A system for doing mathematics by computer, 2nd edition.* [Computer program]. Redwood City, CA: Wolfram Research, Inc.

## APPENDIX A

### Programming Code to Rearrange Fourier Power Spectrum

```
(* The image to be filtered should be cut and pasted to the top of this
program. Using the GRAPH, Convert to Input Form-command,
Mathematica® will convert the image into a two-dimensional matrix.
Assign the matrix a name, and set normft = that name. The output image
is displayed and assigned to the matrix variable inFou.
NOTE: THE IMAGE MATRIX MUST HAVE AN ODD NUMBER
OF ROWS AND COLUMNS FOR THE TRANSFORM TO WORK
PROPERLY!!!*)


(* PASTE IMAGE HERE *)
```

*Follow same procedure as Appendix A to convert image*

```
(* This first section converts the image to the frequency domain by
performing a FFT on the data. Other constants are also assigned here. *)


normft = image;(*change image to your matrix variable
                name *)
norm = normft;
normft = Fourier[normft];
n = Dimensions[norm];
n1 = n[[1]];
n2 = n[[2]];

(* The next two sections convolute the computer's FFT format to place the
zero frequency at the center of the matrix. *)


nhalf1 = Round[n1 / 2 +.1];
nfq = norm;
Do [nfq[[i,j]] = norm[[nhalf1 + i, j]], {i,nhalf1 - 1},{j,n2}];
Do [nfq[[n1 - nhalf1 +i,j]] = norm[[i,j]],{i,nhalf1}, {j,n2}];

nhalf2 = Round[n2 / 2 +.1];
u = nfq;
Do [nfq[[i,j]] = nfq[[i,j + nhalf2]],{i,n1},{j, nhalf2 - 1}];
Do [nfq[[i,n2 - nhalf2 + j]] = u[[i,j]],{i,n1},{j,nhalf2}];

(*Once completed, print the Fourier space image as a 2D surface plot*)

ListPlot3D[nfq]
```

## APPENDIX B

### Low-Pass Spatial Frequency Filter Programming Code

```
(*Low-pass spatial frequency filter: This program takes an
 image and filters all spatial frequencies above a threshold
 frequency that is set by the user. *)

(* The image to be filtered should be cut and pasted to the top of this
program. Using the GRAPH, Convert to Input Form-command,
Mathematica® will convert the image into a two-dimensional matrix.
Assign the matrix a name, and set normft = that name. The output image
is displayed and assigned to the matrix variable inFou.
NOTE: THE IMAGE MATRIX MUST HAVE AN ODD NUMBER
OF ROWS AND COLUMNS FOR THE TRANSFORM TO WORK
PROPERLY!!!*)


(* Highlight the right column, then in the GRAPH menu , Convert to InputForm. This will give a result
shown below: *)


Show[Graphics[{
Raster[{
{0.8421, 0.8375, 0.8358, 0.8319, 0.8327, 0.8311, 0.8319, 0.8295, 0.8304,
0.8292, 0.8267, 0.8241, 0.8272, 0.8260, 0.8245, 0.8189, 0.8248, 0.8162,
0.8166, 0.8127, 0.8131,...},{...,...},{...,...},etc.
{..., 0.7410, 0.7460, 0.7448, 0.7417, 0.7479,
0.7531, 0.7565, 0.7550, 0.7535, 0.7534, 0.7503, 0.7456, 0.7440, 0.7445,
0.7471, 0.7471, 0.7445, 0.7483, 0.7464, 0.7456, 0.7444, 0.7491, 0.7464,
0.7511, 0.7519, 0.7506, 0.7582, 0.7625, 0.7675, 0.7653, 0.7633, 0.7667,
0.7641, 0.7667, 0.7699, 0.7687}},
{{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]


(* Change the format to the following *)
```

1. **Copy this input to the bottom of the file where it will be printed. It is very important to copy this statement; otherwise the aspect ratio and other dimensions will not be proportioned correctly.**

```
 {{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

2. **Delete this statement**

```
Show[Graphics[{
Raster[
```

and

```
 {{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

**3. New statement will look like below**

```
image = {
{0.8421, 0.8375, 0.8358, 0.8319, 0.8327, 0.8311, 0.8319, 0.8295, 0.8304,
0.8292, 0.8267, 0.8241, 0.8272, 0.8260, 0.8245, 0.8189, 0.8248, 0.8162,
0.8166, 0.8127, 0.8131,...},{...,...},{...,...},etc.
{..., 0.7410, 0.7460, 0.7448, 0.7417, 0.7479,
0.7531, 0.7565, 0.7550, 0.7535, 0.7534, 0.7503, 0.7456, 0.7440, 0.7445,
0.7471, 0.7471, 0.7445, 0.7483, 0.7464, 0.7456, 0.7444, 0.7491, 0.7464,
0.7511, 0.7519, 0.7506, 0.7582, 0.7625, 0.7675, 0.7653, 0.7633, 0.7667,
0.7641, 0.7667, 0.7699, 0.7687}};
```

**4. Execute this new statement by highlighting the image array, Command Return**

```
(* minFrq = the threshold frequency for filtering--all frequencies
   equal to or greater than this frequency will be filtered *)

(* This first section converts the image to the frequency domain by
performing a FFT on the data. The amplification factor, threshold
frequency, and other constants are also assigned here. *)


normft = image; (*change image to your matrix variable
                 name *)
normft = Fourier[normft];
n = Dimensions[normft];
n1 = n[[1]];
n2 = n[[2]];
norm =normft;
minFrq =30;

(* The next three Do-loops parse through the image--zeroing out
the high spatial frequencies by the amplification Factor (ampFac)
and leaving the low spatial frequencies alone. *)

Do[norm[[i,j]] = 0,{i,minFrq+1},{j,minFrq+1,n2-minFrq}];
Do[norm[[i,j]] = 0,{i,minFrq+2,n1-minFrq-1},{j,n2}];
Do[norm[[i,j]] = 0,{i,n1-minFrq,n1},
    {j,minFrq+1,n2-minFrq}];


(* Finally perform an inverse FFT to bring the image back and
display it *)
inFou=InverseFourier[norm];
```

```
inFou = Abs[inFou];


Show[Graphics[{
Raster[inFou,
{{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

## APPENDIX C

### High-Pass Spatial Frequency Filter Programming Code

```
(* High-pass spatial frequency filter: This program takes an
image and eliminates all spatial frequencies below a threshold
frequency that is set by the user. *)

(* The image to be filtered should be cut and pasted to the top of this
program. Using the GRAPH, Convert to Input Form-command,
Mathematica® will convert the image into a two-dimensional matrix.
Assign the matrix a name, and set normft = that name. The output image
is displayed and assigned to the matrix variable inFou.
NOTE: THE IMAGE MATRIX MUST HAVE AN ODD NUMBER
OF ROWS AND COLUMNS FOR THE TRANSFORM TO WORK
PROPERLY!!!*)

(* PASTE IMAGE HERE *)

Follow same procedure as Appendix A to convert image

(* maxFrq = the threshold frequency for amplification--all frequencies
equal to or less than this frequency will be amplified *)

(* This first section converts the image to the frequency domain by
performing a FFT on the data. The amplification factor, threshold
frequency, and other constants are also assigned here. *)

normft = image; (*change image to your matrix variable
                  name *)
normft = Fourier[normft];
n = Dimensions[normft];
n1 = n[[1]];
n2 = n[[2]];
norm =normft;
maxFrq =30;


(* The next two sections parse through the image--zeroing out
the low spatial frequencies and leaving the high spatial
frequencies alone. *)


i=1;
While[i <= maxFrq+1,
  Do[norm[[i,j]] = 0,{j,maxFrq+1}];
  Do[norm[[i,j]] = 0,{j,n2-maxFrq+1,n2}];
  i++];

i=n1-maxFrq+1;
```

```
While[i <= n1,
  Do[norm[[i,j]] = 0,{j,maxFrq+1}];
  Do[norm[[i,j]] = 0,{j,n2-maxFrq+1,n2}];
  i++];

(* Finally perform an inverse FFT to bring the image back and
display it *)


inFou=InverseFourier[norm];
inFou = Abs[inFou];


Show[Graphics[{
Raster[inFou,
{{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

## APPENDIX D

### Band-Pass Spatial Frequency Filter Programming Code

```
(* Band-pass spatial frequency filter: This program takes an
image and eliminates all spatial frequencies outside a designated
range of frequencies. *)

(* The image to be filtered should be cut and pasted to the top of this
program. Using the GRAPH, Convert to Input Form-command,
Mathematica® will convert the image into a two-dimensional matrix.
Assign the matrix a name, and set normft = that name. The output image
is displayed and assigned to the matrix variable inFou.
NOTE: THE IMAGE MATRIX MUST HAVE AN ODD NUMBER
OF ROWS AND COLUMNS FOR THE TRANSFORM TO WORK
PROPERLY!!!*)

(* PASTE IMAGE HERE *)
```

***Follow same procedure as Appendix A to convert image***

```
(* minFrq, maxFrq = the lower and upper limits of the band-pass
filter--any frequency outside these limits will be filtered *)

(* This first section converts the image to the frequency domain by
performing a FFT on the data. The upper and lower limits of the
band pass filter and other constants are also assigned here. *)

normft = image; (*change image to your matrix variable name *)
normft = Fourier[normft];
n = Dimensions[normft];
n1 = n[[1]];
n2 = n[[2]];
norm =normft;
minFrq = 4;
maxFrq =30;


(* The next two sections parse through the image--zeroing out
the very low and very high spatial frequencies and leaving the
band pass range alone*)


Do[norm[[i,j]] = 0,{i,minFrq},{j,n2}];
i= minFrq+1
While[i <= maxFrq+1,
  Do[norm[[i,j]] = 0,{j,minFrq}];
  Do[norm[[i,j]] = 0,{j,maxFrq+2, n2-maxFrq}];
  Do[norm[[i,j]] = 0,{j,n2-minFrq+2, n2}];
```

```
  i++];
Do[norm[[i,j]] = 0,{i,maxFrq+2, n1-maxFrq},{j,n2}];
i= n1-maxFrq+1
While[i <= n1-minFrq+1,
  Do[norm[[i,j]] = 0,{j,minFrq}];
  Do[norm[[i,j]] = 0,{j,maxFrq+2, n2-maxFrq}];
  Do[norm[[i,j]] = 0,{j,n2-minFrq+2, n2}];
  i++];
Do[norm[[i,j]] = 0,{i,n1-minFrq+2, n1},{j,n2}];

  (* Finally perform an inverse FFT to bring the image back and
display it *)


inFou=InverseFourier[norm];
inFou = Abs[inFou];


Show[Graphics[{
Raster[inFou,
{{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

## APPENDIX E

### Low Spatial Frequency Amplifier Programming Code

```
(* Low spatial frequency amplifier: This program takes an image and
enhances all spatial frequencies below a threshold frequency that is
set by the user. *)

(* The image to be filtered should be cut and pasted to the top of this
program. Using the GRAPH, Convert to Input Form-command,
Mathematica® will convert the image into a two-dimensional matrix.
Assign the matrix a name, and set normft = that name. The output image
is displayed and assigned to the matrix variable inFou.
NOTE: THE IMAGE MATRIX MUST HAVE AN ODD NUMBER
OF ROWS AND COLUMNS FOR THE TRANSFORM TO WORK
PROPERLY!!!*)


(* PASTE IMAGE HERE *)
```

*Follow same procedure as Appendix A to convert image*

```
(* ampFac = the amplification factor of the low spatial frequencies *)
(* maxFrq = the threshold frequency for amplification--all frequencies
   equal to or less than this frequency will be amplified *)

(* This first section converts the image to the frequency domain by
performing a FFT on the data. The amplification factor, threshold
frequency, and other constants are also assigned here. *)


normft = image; (*change image to your matrix variable
                  name *)
normft = Fourier[normft];
n = Dimensions[normft];
n1 = n[[1]];
n2 = n[[2]];
norm =normft;
ampFac =2;
maxFrq =30;

(* The next two sections parse through the image--multiplying the
low spatial frequencies by the amplification Factor (ampFac) and
leaving the high spatial frequencies alone. *)

i=1;
While[i < = maxFrq+1,
  Do[norm[[i,j]] = norm[[i,j]]*ampFac,{j,maxFrq+1}];
  Do[norm[[i,j]] = norm[[i,j]]*ampFac,{j,n2-maxFrq,n2}];
  i++];
```

```
i=n1-maxFrq;
While[i <=n1,
  Do[norm[[i,j]] = norm[[i,j]]*ampFac,{j,maxFrq+1}];
  Do[norm[[i,j]] = norm[[i,j]]*ampFac,{j,n2-maxFrq,n2}];
  i++];

(* Finally perform an inverse FFT to bring the image back and
display it *)


inFou=InverseFourier[norm];
inFou = Abs[inFou];


Show[Graphics[{
Raster[inFou,
{{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

## APPENDIX F

### High Spatial Frequency Amplifier Programming Code

```
(* High spatial frequency amplifier: This program takes an image and
enhances all spatial frequencies above a threshold frequency that is
set by the user. *)

(* The image to be filtered should be cut and pasted to the top of this
program. Using the GRAPH, Convert to Input Form-command,
Mathematica® will convert the image into a two-dimensional matrix.
Assign the matrix a name, and set normft = that name. The output image
is displayed and assigned to the matrix variable inFou.
NOTE: THE IMAGE MATRIX MUST HAVE AN ODD NUMBER
OF ROWS AND COLUMNS FOR THE TRANSFORM TO WORK
PROPERLY!!!*)
```

**(* PASTE IMAGE HERE *)**

***Follow same procedure as Appendix A to convert image***

```
(* ampFac = the amplification factor of the high spatial frequencies *)
(* minFrq = the threshold frequency for amplification--all frequencies
   equal to or greater than this frequency will be amplified *)

(* This first section converts the image to the frequency domain by
performing a FFT on the data. The amplification factor, threshold
frequency, and other constants are also assigned here. *)


normft = image; (*change image to your matrix variable
                  name *)
normft = Fourier[normft];
n = Dimensions[normft];
n1 = n[[1]];
n2 = n[[2]];
norm =normft;
ampFac =2;
minFrq =30;


(* The next three Do-loops parse through the image--multiplying the
high spatial frequencies by the amplification Factor (ampFac) and
leaving the low spatial frequencies alone. *)


Do[norm[[i,j]] = norm[[i,j]]*ampFac,{i,minFrq+1},
    {j,minFrq+1,n2-minFrq}];
Do[norm[[i,j]] = norm[[i,j]]*ampFac,{i,minFrq+2,n1-minFrq-1},
    {j,n2}];
```

```
Do[norm[[i,j]] = norm[[i,j]]*ampFac,{i,n1-minFrq,n1},
    {j,minFrq+1,n2-minFrq}];


(* Finally perform an inverse FFT to bring the image back and
display it *)


inFou=InverseFourier[norm];
inFou = Abs[inFou];


Show[Graphics[{
Raster[inFou,
{{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

## APPENDIX G

### Oblique Filter Programming Code

### 45° Oblique Filter

```
(* Oblique Filter (45°): This program takes an image and removes
every spatial component that is not oriented along a 45°
slant. It accomplishes this task by performing a FFT on the
image and then eliminating all frequency components that do
not lie on the prime diagonal of the matrix. *)

(* The image to be filtered should be cut and pasted to the top of this
program. Using the GRAPH, Convert to Input Form-command,
Mathematica® will convert the image into a two-dimensional matrix.
Assign the matrix a name, and set normft = that name. The output image
is displayed and assigned to the matrix variable inFou.
NOTE: THE IMAGE MATRIX MUST HAVE AN ODD NUMBER
OF ROWS AND COLUMNS FOR THE TRANSFORM TO WORK
PROPERLY!!!*)


(* PASTE IMAGE HERE *)

Follow same procedure as Appendix A to convert image


(* This first section converts the image to the frequency domain by
performing a FFT on the data. Other constants are also assigned here. *)


normft = image;(*change image to your matrix variable name *)
norm = normft;
norm = Fourier[normft];
n = Dimensions[norm];
n1 = n[[1]];
n2 = n[[2]];
If (EvenQ(n1)

(* The next two sections convolute the computer's FFT format to a more
convenient matrix for oblique filtering. *)


nhalf1 = Round[n1 / 2 +.1];
nfq = norm;
Do [nfq[[i,j]] = norm[[nhalf1 + i, j]], {i,nhalf1 - 1},{j,n2}];
Do [nfq[[n1 - nhalf1 +i,j]] = norm[[i,j]],{i,nhalf1}, {j,n2}];

nhalf2 = Round[n2 / 2 +.1];
u = nfq;
Do [nfq[[i,j]] = nfq[[i,j + nhalf2]],{i,n1},{j, nhalf2 - 1}]
Do [nfq[[i,n2 - nhalf2 + j]] = u[[i,j]],{i,n1},{j,nhalf2}];
```

```
(* Now eliminate all nondiagonal elements. Also, because matrices are not necessarily square, eliminate
extra rectangular components. *)


Off[General::spelli];
minDim = Min[n1, n2];
(* Prime Diagonal will be offset if the matrix is rectangular, use the
   constant term (zero frequency) to determine the center of the matrix *)
offsetI = nhalf1 - Round[minDim/2 + 0.1];
offsetJ = nhalf2 - Round[minDim/2 + 0.1];
Do [If [i != j, nfq[[i+offsetI,j+offsetJ]] = 0],{i,minDim},{j,minDim}];
If [n1 > minDim, (* The matrix was rectangular with too many rows, so
delete extraneous rows *)
  Do [nfq[[i,j]] = 0, {i, offsetI},{j,n2}];
  Do [nfq[[i,j]] = 0, {i, offsetI+minDim+1, n1},{j, n2}];
  ];
If [n2 > minDim, (* Matrix has too many columns *)
  Do [nfq[[i,j]] = 0, {i, n1},{j, offsetJ}];
  Do [nfq[[i,j]] = 0, {i, n1},{j, offsetJ+minDim+1, n2}];
  ];


(* Nfq is now properly filtered. We must now send it through a manipulation
to put the matrix back in the Fourier format the Apple understands.*)


ob = nfq;
i = 1;
While [i <= n1 ,
   If [(nhalf1 +i - 1) <= n1, x = nhalf1 + i - 1, x = i - nhalf1];
   j = 1;
   While [ j <= n2 ,
     If [(nhalf2 + j - 1) <= n2 , y = nhalf2 + j - 1, y = j - nhalf2];
     ob[[i,j]] = nfq[[x,y]];
     j++];
   i++];


(* Finally perform an inverse FFT to bring the image (ob) back and
display it *)


inFou = InverseFourier[ob];


inFou = Abs[pictObliq];
Show[Graphics[{
{{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

## 135° Oblique Filter

(* Oblique Filter (135°) : This program takes an image and removes every spatial component that isn't oriented along a 135° slant. It accomplishes this task by performing a FFT on the image and then eliminating all frequency components that do not lie on the diagonal of the matrix that runs from the lower left to upper right. *)

(* The image to be filtered should be cut and pasted to the top of this program. Using the GRAPH, Convert to Input Form-command, Mathematica® will convert the image into a two-dimensional matrix. Assign the matrix a name, and set normft = that name. The output image is displayed and assigned to the matrix variable inFou.
NOTE: THE IMAGE MATRIX MUST HAVE AN ODD NUMBER OF ROWS AND COLUMNS FOR THE TRANSFORM TO WORK PROPERLY!!!*)

**(* PASTE IMAGE HERE *)**

***Follow same procedure as Appendix A to convert image***

(* This first section converts the image to the frequency domain by performing a FFT on the data. Other constants are also assigned here. *)

```
normft = image;(*change image to your matrix variablename*)
norm = Fourier[normft];
n = Dimensions[norm];
n1 = n[[1]];
n2 = n[[2]];
```

(* The next two sections convolute the computer's FFT format to a more convenient matrix for oblique filtering. *)

```
nhalf1 = Round[n1 / 2 +.1];
nfq = norm;
Do [nfq[[i,j]] = norm[[nhalf1 + i, j]], {i,nhalf1 - 1},{j,n2}];
Do [nfq[[n1 - nhalf1 +i,j]] = norm[[i,j]],{i,nhalf1}, {j,n2}];

nhalf2 = Round[n2 / 2 +.1];
u = nfq;
Do [nfq[[i,j]] = nfq[[i,j + nhalf2]],{i,n1},{j, nhalf2 - 1}]
Do [nfq[[i,n2 - nhalf2 + j]] = u[[i,j]],{i,n1},{j,nhalf2}];
```

(* Now eliminate all nondiagonal elements. Also, because matrices are not necessarily square, eliminate extra rectangular components. *)

```
Off[General::spelli];
```

```
minDim = Min[n1, n2];
(* Prime Diagonal will be offset if the matrix is rectangular,
 use the constant term (zero frequency) to determine the center
 of the matrix *)
offsetI = nhalf1 - Round[minDim/2 + 0.1];
offsetJ = nhalf2 - Round[minDim/2 + 0.1];
Do [If [i + j != minDim+1, nfq[[i+offsetI,j+offsetJ]] = 0],
   {i,minDim},{j,minDim}];
If [n1 > minDim, (* The matrix was rectangular with too many
  rows, so delete extraneous rows *)
  Do [nfq[[i,j]] = 0, {i, offsetI},{j,n2}];
  Do [nfq[[i,j]] = 0, {i, offsetI+minDim+1, n1},{j, n2}];
  ];
If [n2 > minDim, (* Matrix has too many columns *)
  Do [nfq[[i,j]] = 0, {i, n1},{j, offsetJ}];
  Do [nfq[[i,j]] = 0, {i, n1},{j, offsetJ+minDim+1, n2}];
  ];


(* Nfq is now properly filtered.  We must now send it through a manipulation
to put the matrix back in the Fourier format the Apple understands.*)

ob = nfq;
i = 1;
While [i <= n1 ,
  If [(nhalf1 +i - 1) <= n1, x = nhalf1 + i - 1, x = i - nhalf1];
  j = 1;
  While [ j <= n2 ,
    If [(nhalf2 + j - 1) <= n2 , y = nhalf2 + j - 1,
          y = j - nhalf2];
    ob[[i,j]] = nfq[[x,y]];
    j++];
  i++];


(*  Finally perform an inverse FFT to bring the image (ob) back and
display it *)


inFou = InverseFourier[ob];


inFou = Abs[pictObliq];
Show[Graphics[{
{{0.000000, 0.000000}, {1.000000, 0.492441}}]
}], AspectRatio->0.49244,
PlotRange->{{0.00000, 1.00000}, {0.00000, 0.49244}}]
```

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | Jan 1994 | |

**4. TITLE AND SUBTITLE**
The Sharper Image: Implementing a Fast Fourier Transform (FFT) to Enhance a Video-Captured Image

**5. FUNDING NUMBERS**
62233N
MM33130.009-7305

**6. AUTHOR(S)**
J. E. Parker, W. K. Krebs, J. S. Marsh[1], D. L. Still, and L. A. Temme

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
NAVAEROMEDRSCHLAB
51 HOVEY ROAD
PENSACOLA FL 32508-1046

**8. PERFORMING ORGANIZATION REPORT NUMBER**
NAMRL Special Report 94-1

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Naval Medical Research and Development Command
National Naval Medical Center, Bldg. 1
Bethesda, MD 20889-5606

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
[1]Department of Physics, University of West Florida, Pensacola, FL 32514

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

The lack of visual information and cues in night vision images has produced a history of in-flight mishaps where night vision devices (NVDS) were used as the primary source of sight. In an effort to produce improved night vision images and thereby reduce in-flight mishaps, a video enhancement technique, the sharper image, was developed to enhance, filter, and study images generated by NVDS. The sharper image technique enhances night vision images by filtering or amplifying the spatial frequencies that are distorted by NVDS. In this report, we describe the necessary methodology and procedures to enhance, filter, or process any visual image. We also discuss other computational image software and review the advantages and disadvantages of each.

**14. SUBJECT TERMS**
Image enhancement, Filtering, FFT, Discrete fourier transfer, Spatial frequency component, Night vision, Image capture

**15. NUMBER OF PAGES**
35

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | SAR |